

Задача А. Платные дороги

Имя входного файла: `highways.in`
Имя выходного файла: `highways.out`
Ограничение по времени: 3 секунды
Ограничение по памяти: 256 мегабайт

Мэр одного большого города решил ввести плату за проезд по шоссе, проходящим в районе города, чтобы снизить объем транзитного транспорта. В районе города проходит n шоссе.

Но руководство области, в которой расположен город, воспротивилось планам мэра. Действительно — дальнобойщики представляют собой неплохой источник доходов для большого количества кафе и гостиниц в небольших городках.

В результате решили, что плата будет введена только на шоссе, которые проходят через город.

В городе используется развитая система метрополитена, всего в городе есть m станций метро. Решено было, что шоссе проходит через город, если либо одна из станций метро расположена непосредственно на шоссе, либо есть хотя бы одна станция с каждой стороны от шоссе.

Помогите теперь мэру определить, какие шоссе проходят через город.

Формат входных данных

Первая строка входного файла содержит два целых числа: n и m — количество шоссе и количество станций метро, соответственно ($1 \leq n, m \leq 100\,000$).

Следующие n строк описывают шоссе. Каждое шоссе описывается тремя целыми числами a , b и c и представляет собой прямую на плоскости, задаваемую уравнением $ax + by + c = 0$ ($|a|, |b|, |c| \leq 10^9$).

Следующие m строк входного файла описывают станции метро. Каждая станция описывается двумя целыми числами x и y и представляет собой точку на плоскости с координатами (x, y) ($|x|, |y| \leq 10^9$).

Формат выходных данных

Первая строка выходного файла должна содержать одно целое число — количество шоссе, которые проходят через город. Вторая строка должна содержать номера этих шоссе в возрастающем порядке. Шоссе нумеруются от 1 до n в порядке, в котором они описаны во входном файле.

Примеры

highways.in	highways.out
4 2 0 1 0 1 0 1 1 1 0 1 1 -1 0 0 2 0	3 1 3 4

Задача В. Соединение и разъединение

Имя входного файла: `connect.in`
Имя выходного файла: `connect.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Вы когда-нибудь слышали про обход в глубину? Например, используя этот алгоритм, вы можете проверить является ли граф связным за время $O(E)$. Вы можете даже посчитать количество компонент связности за то же время.

А вы когда-нибудь слышали про систему непересекающихся множеств? Используя эту структуру, вы можете быстро обрабатывать запросы “Добавить ребро в граф” и “Посчитать количество компонент связности в графе”.

А вы когда-нибудь слышали о *динамической* задаче связности? В этой задаче вам необходимо обрабатывать три типа запросов:

1. Добавить ребро в граф.
2. Удалить ребро из графа.
3. Посчитать количество компонент связности в графе.

Можно считать, что граф является неориентированным. Изначально граф является пустым.

Формат входных данных

В первой строке находятся два целых числа N и K — количество вершин и количество запросов, соответственно ($1 \leq N \leq 300\,000$, $0 \leq K \leq 300\,000$). Следующие K строк содержат запросы, по одному в строке. Каждый запрос имеет один из трех типов:

1. $+ \ u \ v$: Добавить ребро между вершинами u и v . Гарантируется, что такого ребра нет.
2. $- \ u \ v$: Удалить ребро между u и v . Гарантируется, что такое ребро есть.
3. $?$: Посчитать количество компонент связности в графе.

Вершины пронумерованы целыми числами от 1 до N . Во всех запросах $u \neq v$.

Формат выходных данных

Для каждого запроса типа ‘?’, Выведите количество компонент связности в момент запроса.

Примеры

<code>connect.in</code>	<code>connect.out</code>
5 11	5
?	1
+ 1 2	1
+ 2 3	2
+ 3 4	
+ 4 5	
+ 5 1	
?	
- 2 3	
?	
- 4 5	
?	

Задача С. Покраска забора

Имя входного файла: `paint.in`
Имя выходного файла: `paint.out`
Ограничение по времени: 0.7 секунда
Ограничение по памяти: 256 мегабайт

Мэр города Многоярославца решил построить перед своим домом забор из n деревянных досок и нанять лучшего маляра города для его покраски. Поскольку забор должен стать главной достопримечательностью города, лучший дизайнер города для каждой доски назначил тщательно выбранный цвет, в который она должна быть покрашена.

Для покраски главный маляр решил применить новейшую технологию, специально разработанную им для выполнения этого задания. Покраской забора будет заниматься специальный робот, который за один час может покрасить произвольный отрезок забора (набор соседних досок) в некоторый цвет. Поскольку задание должно быть выполнено как можно быстрее, требуется составить программу для робота, которая позволит достичь требуемой раскраски за минимальное время. Оставить какую-то из досок непокрашенной, естественно, запрещается.

Формат входных данных

В первой строке входного файла записано число n ($1 \leq n \leq 300$), где n количество досок в заборе. Вторая строка содержит строку из n символов, описывающую требуемую покраску забора. Цвета обозначаются заглавными латинскими буквами.

Формат выходных данных

В первой строке выходного файла выведите m - наименьшее возможное время покраски забора в часах. Следующие m строк должны содержать программу покраски для робота. Каждая строка должна содержать два числа l_i и r_i , а также заглавную букву латинского алфавита, задающую цвет c_i и означает, что робот должен покрасить участок забора с l_i по r_i - доску в цвет c_i (если длина забора n , должно выполняться $1 \leq l_i \leq r_i \leq n$).

Примеры

paint.in	paint.out
5 ABVCA	3 1 5 A 4 4 C 2 3 B
2 AA	1 1 2 A

Задача D. 17 стульев

Имя входного файла: `trader.in`
Имя выходного файла: `trader.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

Остап Бендер снова пытается получить причитающиеся драгоценности, но на этот раз они были заперты в шкатулке, для открытия которой необходимо иметь N ключей. По закономерной случайности каждый из ключей был спрятан в одном из N стульев, распроданных на недавнем аукционе. После аукциона эти стулья были развезены в N городов.

И вот теперь Остап решил на новую безумную затею: заехать в каждый из городов и, провернув в каждом из них аферу, выкрасть необходимые ключи. Чтобы избежать конфликтов с недоброжелателями, Остап не хочет больше одного раза появляться в каком-либо городе. Также у Остапа есть список цен за проезд между каждой парой городов. Изначально Остап находится в городе под номером 1 и после посещения всех городов может незаметно скрыться из этой страны.

Помогите Остапу найти порядок посещения городов, при котором ему потребуется потратить как можно меньше средств на странствия, и тогда, возможно, он поделится с Вами добытыми бриллиантами.

Формат входных данных

Первая строка содержит единственное число N — количество городов ($1 \leq N \leq 17$).

Следующие N строк содержат по N целых неотрицательных чисел. j -тое число в i -той строке означает стоимость проезда из города i в город j ($0 \leq a_{ij} \leq 100$). Если $a_{ij} > 0$, то проезд стоит a_{ij} рублей, иначе — это означает, что из города i в j невозможно проехать напрямую.

Формат выходных данных

В первой строке выведите минимальную сумму денег, необходимую для посещения всех городов Остапом. В следующей строке выведите N чисел — порядок посещения городов, при котором эта сумма достигается. Если затею Остапа невозможно вывести, то в единственной строке выходного файла выведите число -1.

Примеры

<code>trader.in</code>	<code>trader.out</code>
3 0 3 2 3 0 6 2 6 0	8 1 3 2
5 0 6 4 0 0 6 0 7 0 7 4 7 0 0 0 0 0 0 0 2 0 7 0 2 0	20 1 3 2 5 4